ON GLOBAL FEATURE POOLING FOR FINE-GRAINED VISUAL CATEGORIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Global feature pooling is a modern variant of feature pooling providing better interpretatability and regularization. Although alternative pooling methods exist (*e.g.* max, l_p norm, stochastic), the averaging operation is still the dominating global pooling scheme in popular models. As fine-grained recognition requires learning subtle, discriminative features, is average pooling still the optimal strategy? We first ask: "is there a difference between features learned by global average and max pooling?" Visualization and quantitative analysis show that max pooling encourages learning features of different spatial scales. We then ask "is there a single global feature pooling variant that's most suitable for fine-grained recognition?" A thorough evaluation of nine representative pooling algorithms finds that: max pooling outperforms average pooling consistently across models, datasets, and image resolutions; it does so by reducing the generalization gap; and generalized pooling's performance increases almost monotonically as it changes from average to max. We finally ask: "what's the best way to combine two heterogeneous pooling schemes?" Common strategies struggle because of potential gradient conflict but the "freeze-and-train" trick works best. We also find that post-global batch normalization helps with faster convergence and improves model performance consistently.

1 INTRODUCTION

Deeply rooted in the works of complex cells in the visual cortex (Hubel & Wiesel, 1962) and locally orderless images (Koenderink & Van Doorn, 1999), feature pooling has been an indispensable component of visual recognition in both traditional bag-of-words (BOW) frameworks (Csurka et al., 2004; Lazebnik et al., 2006) using hand-crafted features (*e.g.* SIFT (Lowe, 2004), HOG (Dalal & Triggs, 2005)), and modern convolutional neural networks (CNNs) (LeCun et al., 1998; Krizhevsky et al., 2012). A recent variant of this technique, called "global feature pooling" (Lin et al., 2013), distinguishes itself by defining its pooling kernel the same size as input feature map. The pooling output is a scalar value indicating the existence of certain features (or patterns). Benefits of global pooling are two-fold: allowing better interpretation of the underlying filters as feature detectors, and serving as a strong network regularizer to reduce overfitting. Global pooling is thus used in most, if not all, recent state-of-the-art deep models (Szegedy et al., 2015; He et al., 2016; Szegedy et al., 2017; Huang et al., 2017; Hu et al., 2018) in visual recognition. Unless otherwise noted, all the pooling methods discussed in this paper are used as the global pooling layer.

Feature pooling is also of special interests to Fine-grained Visual Categorization (FGVC) (Rosch et al., 1976; Nilsback & Zisserman, 2010; Farrell et al., 2011), where objects are classified into subcategories rather than basic categories. Carefully designed pooling schemes can learn helpful discriminative features and yield better performance without requiring more conv-layers in the network. Wang et al. (2018) provided a good example that combines three pooling operations: average, max and cross-channel pooling to learn to capture class-specific discriminative patches. Another major research direction is higher-order pooling: Lin et al. (2015) proposed to apply bilinear pooling (also know as second-order pooling) to capture pairwise correlations between the feature channels and model part-feature interactions; Gao et al. (2016) proposed compact bilinear pooling that applies random maclaurin projection and tensor sketch projection to approximate the outer product operation, greatly reducing parameters without sacrificing accuracy; Works along this line of research include low-rank bilinear pooling (Kim et al., 2016), grassmann pooling (Wei et al., 2018),

kernel pooling (Cui et al., 2017), and Alpha-pooling Simon et al. (2017), *etc.* Although higher-order pooling methods output a vector rather than a scalar, they're still relevant as they reside in the same location as the global pooling layer.

The most common pooling operations are average, max and striding. Striding always takes the activation at a fixed location, thus is never applied as global pooling. An abundant set of pooling flavors exist for both traditional and modern feature extractors. Stochastic pooling (Zeiler & Fergus, 2013) randomly chooses an activation according to a multinomial distribution decided by activation strength in the pooling region. Fractional max pooling (Graham, 2014) can be adapted to fractional sized pooling regions. Spatial pyramid pooling (He et al., 2015) outputs the combination of multiple max pooling with different sized pooling kernels. S3Pool (Zhai et al., 2017), or stochastic spatial sampling Pooling, randomly picks a sub-region to apply max pooling to. Detail-preserving pooling (Saeedan et al., 2018) computes the output as the linear combination of input feature pixels whose weight is proportional to differences of the input intensities. Translation invariant pooling (Zhang, 2019) borrowed the idea of anti-alias by low-pass filtering from signal processing. A major pooling family, generalized pooling, aims to find a smooth *transition* between average and max pooling: k-max pooling (Kalchbrenner et al., 2014) outputs the average of the k highest activations of the feature map; l_p norm pooling generalizes pooling to the p-norm of the input feature map (Boureau et al., 2010); soft pooling (Boureau et al., 2010), or softmax pooling, outputs the sum of feature map weighted by softmax output; mixed pooling (Lee et al., 2016) computes a weighted sum of the max and average pooling; gated pooling (Lee et al., 2016) is similar to mixed pooling but the weight is learned instead. To the best of our knowledge, these pooling operations remain largely unexplored in the global pooling scenario.

An interesting observation is that all highly-ranked classification models "happen" to choose the same *averaging operation* in their global pooling layer. Is this an arbitrary choice or actually the optimal strategy? How does average pooling compare against the other pooling schemes (*e.g.* max) in general image classification and also fine-grained visual recognition? Research (Boureau et al., 2010; Murray & Perronnin, 2014; Scherer et al., 2010; Hu et al., 2018; Zhou et al., 2016) has shown that the selection of feature pooling affects the algorithm's performance, whether using hand-crafted features or deep features. Specially, Murray & Perronnin (2014) showed max pooling has superior performance in the traditional recognition framework because of its better pattern discriminability, and the same conclusion was made by an experimental evaluation of Scherer et al. (2010) using LeNet-5 (LeCun et al., 1998) on the Caltech 101 (Fei-Fei et al., 2007) and NORB (Jarrett et al., 2009) dataset. Boureau et al. (2010) provided a theoretical proof that "max pooling is particularly well suited to the separation of features that are very sparse." However, in squeeze and excitation networks (Hu et al., 2018), global max pooling is reported to achieve 0.29% higher top-1 error and 0.05% higher top-5 error than average pooling. Similar results were reported by Zhou et al. (2016) using VGG (Simonyan & Zisserman, 2015) and GoogleNet (Szegedy et al., 2016). It seems max pooling is less preferred as a global pooling scheme than before. These intriguing contrasts call for a careful examination of both pooling schemes.

Our investigation begins with the two most common global average and max pooling. Specially, we're interested to know what features have both pooling methods helped learned. *Feature map* visualization indicates that max pooling produces *sparser* final conv-layer feature maps. This is further verified quantitatively by two perceptually-consistent sparsity metrics: discrete entropy and thresholded l_0 norm. Visualization of final conv-layer *filters* further helps us conclude empirically that: *global average pooling encourages object-level features while global max pooling focuses more on part-level features.* As class-specific features often reside in localized object parts in fine-grained datasets, it's equal to say global max pooling find more discriminative features, well aligned with previous findings (Murray & Perronnin, 2014; Zhou et al., 2016).

The second question to answer is that "is there a single optimal pooling operation on different fine-grained datasets across different models?" We evaluate *nine* representative pooling schemes, which are: average, max, k-max, l_p norm, soft, logavgexp, mixed, gated, and stochastic pooling, in the experiment section. We make several observations: max pooling outperforms average pooling across datasets, input resolutions, and models. The reason behind this phenomenon, besides their feature differences, is relevant to the fact that *max pooling generalizes better*. Most pooling methods we evaluated performs better than average pooling, with k-max ($k = 2$) and mixed pooling ($\alpha = 0.5$) being the top two. Our k-max pooling model, when trained properly, beats all previous higher-order pooling methods using the same backbone. The fact that no single pooling works best for all models

leads to the need for learnable pooling, where the pooling function is not chosen by heuristic, but optimized via gradient descent. However, our finding that model performance decrease and generalization gap increases in an almost monotonic way when generalized pooling changes from max to average casts a shadow upon the learnable generalized pooling. A pooling is better not because it minimizes training loss, but because it better regularizes the model. Throughout our experiment, *post-global batch normalization* is applied as another key ingredient achieving consistent performance improvement and faster convergence.

Finally, we explore the integration of heterogeneous pooling. Since different features can be learned by average or max pooling, our assumption is that learning a model with heterogeneous poolings will lead to better performance, but what's the best way to integrate them? We review and evaluate three common strategies, but found their improvement upon single pooling is limied. Our hypothesis is that different pooling methods interfere and cancel each other out when learned together. We instead propose to apply the "freeze-and-train" trick. The intuition is that the frozen branch won't degrade during training and the gradients will be well separated. The resulting architecture only adds a tiny amount of parameters to a backbone network, but consistently outperforms single pooling models.

2 POOLING SCHEMES OVERVIEW

Here we describe in detail the pooling algorithms we evaluate in the experiment section. We use the following notations: the input feature map is a 3D Tensor: $\boldsymbol{X} \in \mathbb{R}^{c \times h \times w}$, with c, h, w being the channel size, height and width. Each feature map is a 2D matrix, but in most cases the algorithm cares less about the 2D structure than individual activation strength, so we simply use its flattened vector form $\vec{x}$. The scalar output of each feature map is denoted as y. We use x_i to index individual elements of $\vec{x}$ and set $N = hw$ as the feature map size.

Average pooling can be expressed as:

$$y = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{1}$$

Max pooling picks the maximum value of $\vec{x}$:

$$y = x_i, i = \text{argmax}(\vec{x}) \tag{2}$$

Stochastic pooling chooses a single activation from the input based on a multinomial distribution decided by relative activation strength.

$$y = x_i, i \sim P(p_1, ..., p_i, ..., p_N), \tag{3}$$

where each probability p_i is given by $p_i = \frac{x_i}{\sum_{j=1}^{N} x_j}$. We can safely assume $x_i > 0$ holds for all i as the input is always the output of a ReLU layer.

L_p **norm pooling** computes the L_p norm:

$$y = (\frac{1}{N} \sum_{i} x_i^p)^{(\frac{1}{p})} \tag{4}$$

When $p = 1$, l_p norm reduces to average pooling and when $p \to \infty$, l_p norm approaches max pooling.

Soft pooling, or softmax pooling, reweights the inputs as:

$$y = \sum_{i} \frac{\exp(\beta x_i)}{\sum_j \exp(\beta x_j)} x_i \tag{5}$$

When $\beta = 0$, soft pooling equals average pooling, and when $\beta \to \infty$, soft pooling behaves like max pooling.

Logavgexp pooling applies three functions consecutively to the input:

$$y = \frac{1}{\beta} \log \frac{1}{N} \sum_{i} \exp(\beta x_i) \tag{6}$$

Logavgexp is average pooling when $\beta = 0$ and max pooling when $\beta \to \infty$.

K-max pooling averages the top k activation:

$$y = \frac{1}{K} \sum_i x_i, i \in \text{argmax}_K(x) \tag{7}$$

With a slight abuse of notation, we use argmax_K to return the indices of the top k activations. It is clear that when $k = 1$, k-max pooling is the same as max pooling, and when $k = N$, k-max pooling is average pooling.

Mixed pooling combines max and average pooling:

$$y = \alpha f_{\text{max}}(x) + (1 - \alpha) f_{\text{avg}}(x) \tag{8}$$

When $\alpha = 0$, mixed pooling is average pooling, and when $\alpha = 1$, mixed pooling becomes max pooling. Note α is not learned in this case.

Gated pooling learns a universal weight for combining max and averagioffe2015batche pooling:

$$y = \sigma(w^T x) f_{\text{max}}(x) + (1 - \sigma(w^T x)) f_{\text{avg}}(x), \tag{9}$$

where $\sigma(w^T x) = \frac{1}{1+\exp(-w^T x)}$ is the sigmoid function.

3 Average or Max? A Filter-level Comparison

In this section, we aim to answer the question: "what's the difference between features learned by max and average pooling?". Our explorations are partially influenced by works in network visualization (Zeiler & Fergus, 2013; Zhou & Lin, 2016). We visualize the final-layer feature maps and filters to show the pattern differences of networks with max and average pooling (section 3.1 and 3.2): global average pooling encourages object-level features and global max pooling focuses on part-level features.

Difference in Feature Maps - Feature maps are nice indicators of the existence of certain patterns detected by corresponding filters. Final conv-layer features maps are of special interest as they are directly connected to the global pooling layer. We visualize the final-layer *feature maps* of a neural network trained with global average and max pooling. The feature maps are arranged in a grid and the activations are scaled to [0,1] by min/max normalization for better visualization. Examples of feature map visualization from both pooling methods are shown on the right side of Figure 1 (white borders are added for better visualization). The feature maps of the max-pooling model are visually much sparser, with sharply peaked activations. Features maps from the average-pooling model, on the other hand, are more intensively activated. The same phenomenon is observed across multiple images.

Quantitative Analysis - The formal way to measure the sparsity of a feature map is through its l_0 norm, which counts the number of non-zero entries in the matrix. Entropy is also used to measure sparsity, where low entropy indicates high sparsity. But both metrics struggle with noisy or disorderly data. A perceptually sparse feature map may have high l_0 norm if all pixels have a small non-zero value, while two feature maps with visually different sparsity may have very similar entropy.

To better reflect perceptual sparsity and allow for better noise tolerance, we propose a modified version of each of the above metrics, denoted as $S^{\{1,2\}}$. Suppose the evaluation dataset contains n images and there are m filters (channels) in the final conv-layer. Each feature map is normalized to be a valid probability distribution (using softmax or l_1 norm, *etc.*). Both of the proposed metrics first compute the sparsity of each individual feature map $s^{\{1,2\}}$, and take the average across channels and images: $S^{\{1,2\}} = \frac{1}{nm} \sum_{j=0}^{n} \sum_{i=0}^{m} s_{ij}^{\{1,2\}}$

The first metric is *discrete entropy*: each feature map is represented as a probability distribution using a normalized histogram with k bins. The probability of each bin is p_i, such that $\sum_i p_i = 1$. Discrete entropy preserves most information in a probability distribution but discards small perturbations

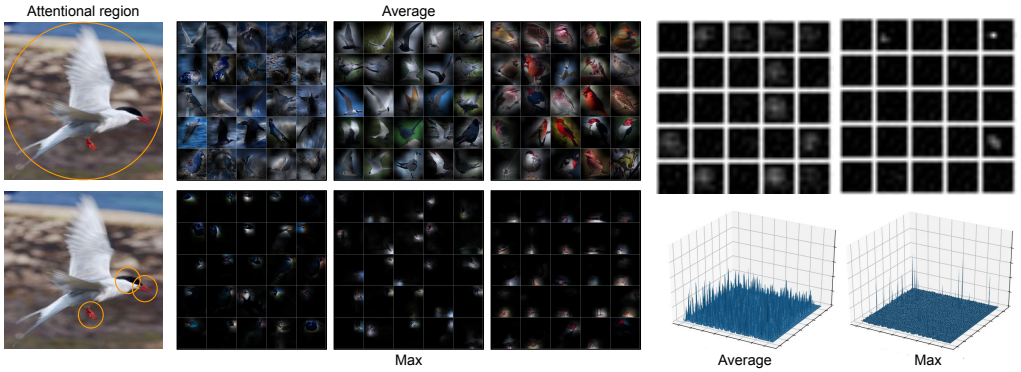

Figure 1: First column: attentional regions indicated by filter patterns. Second to fourth column: filter visualization by maximum activated images. Last two columns: 2D (top row) and 3D (bottom row, height indicating activation strength) feature map visualization. It's clear that max pooling produces sparser feature maps and encourages part-level features to be learned.

through histogram bin quantization. The discrete entropy s^1 of each feature map is computed as: $s^1 = \sum_{i=0}^{k} -p_i log(p_i)$

The second metric is *thresholded* l_0*-norm*. Consider each feature map as a 1D vector. Let the feature map value at location i be x_i and the feature map size be N. When the threshold is t, the thresholded l_0 norm s^2 for each feature map is defined as: $s^2 = \sum_{i=0}^{N} \mathbf{1}(x_i >= t)$ where $\mathbf{1}(x_i >= t)$ is an indicator function that returns 1 if $x_i >= t$ and 0 otherwise. Small positive values are suppressed by thresholded l_0-norm.

Besides global average and max pooling, we're also interested in the sparsity changes during the transition between them. Our assumption is that the feature map sparsity will decrease as a generalized pooling transforms from max to average pooling, which will be verified in Section 5.1.

Difference in Filters - Although different methods to understand a filter's learned pattern exist in the network interpretation literature, one simple and accurate method is to apply the up-sampled feature map as a heatmap (or mask) on top of the input image and observe the patterns in the highlighted regions. To better visualize a given filter, we rank all the input images in descending order according to their activation strength after the global pooling layer. The images causing the highest activations are selected as representatives of each filter's activation pattern. We show a representative filter visualization example on the left side of Figure 1. The difference in learned features between the two pooling methods is clear: the global average pooling features are mostly object-level along with background patterns, *e.g.* white-colored birds, sky, grass, water, *etc.* Global max pooling, on the other hand, focuses on highly-localized object parts, *e.g.* eyes, legs, wing bars, *etc.* In fine-grained datasets, discriminative features often reside in localized parts. The global max pooling's attention to part-level features suggests its ability to find highly discriminative, class-specific features, which aligns well with previous findings (Boureau et al., 2010; Murray & Perronnin, 2014).

4 Heterogeneous Pooling Integration

As we've shown that global average and max pooling learn features of different spatial scales, a natural question arises: "does the combination of heterogereous pooling methods lead to better performance?". In this section, we review an offline training baseline and several end-to-end learnable methods.

We first consider a baseline method treating CNNs as feature extractors (Sharif Razavian et al., 2014): concatenating the features from max and average pooling models and training an offline classifier using the concatenated features. Although simple in concept and easy to implement, this offline training method has twice as many parameters and is not end-to-end learnable.

Mixed pooling (Lee et al., 2016) is our first end-to-end learning strategy: the output of the two pooling functions are combined by a weighted sum. The second strategy is "channel split": splitting

the final conv-layer channels and applying one type of pooling to each split. The third strategy, called "branching", is to add a separate linear layer for each pooling method (Boureau et al., 2010; He et al., 2015; Wang et al., 2018). Mixed pooling and channel split both add one hyper-parameter: α and split ratio. The branching strategy adds a constant number of parameters for each pooling scheme.

We will show that the above end-to-end learnable models provide *little* improvement over single-pooling models. Our hypothesis is that lower layer filters become confused by the gradient signals coming from the two different pooling schemes. We propose a simple modification to the branching strategy to separate the gradient flow of different pooling methods using the "freeze-and-train" trick: the backbone model with global average pooling layer is trained until convergence;then the weights are frozen and a new linear layer with global max pooling is added. The newly added linear layer is then trained until convergence. Finally, the whole network is trained together. Experimental results are provided in Section 5.3.

5 EXPERIMENTS

We use three public fine-grained datasets in our experiments: CUB-200-2011 (Birds) with 200 classes, 5,994 images for training and 5,794 for testing; Stanford Cars (Cars) with 196 classes, 8,144 images for training and 8,041 for testing; and FGVC Aircraft (Aircraft) with 100 classes, 6,667 images for training and 3,333 for testing. We use ResNet-50 as the backbone model for most experiments and Inception-v3 in the pooling benchmark experiment. We use stochastic gradient descent (SGD) with universal learning rate schedule for ResNet-50 models. We use the Adam optimizer for Inception-v3. Resnet-50 models are trained with learning rate 10^{-3} for 30 epochs, and 10^{-4} for 20 more epochs in most cases. Inception-v3 models are trained with learning rate 10^{-4} for 30 epochs and 10^{-5} for 20 epochs. All models converge under the described settings. All models are fine-tuned from models pretrained on Imagenet (Deng et al., 2009). Standard image enhancement including horizontal flipping, random cropping and resizing, and color normalization are used during training. All models are trained with batch size of 16, weight decay 10^{-4} and momentum 0.9. The input image size is 448×448 in most cases, aligned with previous works (Lin et al., 2015; Cui et al., 2017). Testing is always performed on the center-crop, *without* multi-crop averaging.

Train from scratch vs. fine-tune - To make a fair comparison between all pooling methods, models should be trained from scratch. However, as fine-grained datasets are usually small in size, most current models are fine-tuned from pretrained models to get better results. Training from scratch requires careful learning rate scheduling for each algorithm, and an exhaustive hyperparameter search is beyond the scope of this work. Training from scratch also takes much longer to converge. We acknowledge that the fine-tuned models may be heavily influenced by the inductive bias present in the pretrained weights, which may cause models to have similar performance. Nevertheless, we are still able to make some interesting observations which we assume will be generally applicable.

5.1 PERCEPTUAL SPARSITY FOR K-MAX POOLING

We now aim to discover how perceptual sparsity metrics (discrete entropy and thresholded l_0 norm) change with respect to different k values in k-max pooling. The input size is 448×448, leading to a final feature map size is $14 \times 14 = 196$, so $k \in \{1, \ldots, 196\}$. We sample 10 evenly spaces points $\{1, 14, 27, ..., 196\}$ (recall that k-max pooling is max pooling when k=1 and average pooling when k=196). We train on the Birds dataset. Results are shown on the left side and middle of Figure 2.

The number of bins in the histogram is a hyperparameter for discrete entropy. We report results for multiple bin numbers $\{4, 8, 16\}$. The discrete entropy computed using different bin numbers varies, but we observe a roughly monotonic increase as k grows from 1 to 196 universally. Specifically, max pooling (k=1) gets 0.018, 0.095, 0.34 and average pooling gets 0.10, 0.40, 0.91 respectively. The gaps between them are 0.082, 0.305, 0.57.

For thresholded l_0 norm, we select three values for t: 0.01, 0.015, and 0.02. The thresholded l_0 norms also increase nearly monotonically with increasing k. Max pooling (k=1) gets 4, 3 and 1 and average pooling (k=196) gets 15, 8, 6. The gaps are 11, 5, and 5.

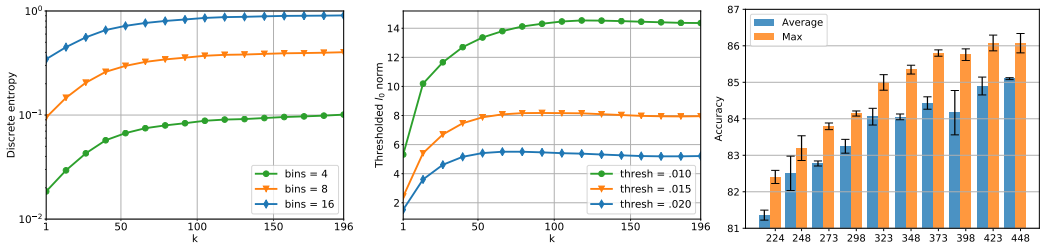

Figure 2: Left and middle: discrete entropy (plotted in log scale) and thresholed l_0 norm for varied k in k-max pooling. Feature maps become more densly activated when k-max pooling transforms from max to average pooling (from k=1 to k=196). Right: model accuracy against input resolutions. Max pooling is consistently better than average.

As lower discrete entropy and lower thresholded l_0 norm indicate higher sparsity of the final conv-layer features, our assumption is validated that max pooling encourages sparser final-conv layer feature maps and the sparsity decreases monotonically as max pooling is transformed to average pooling.

5.2 Global Pooling Evaluation

In this section we present the evaluation results for nine global pooling schemes on three public datasets. We discuss how different poolings compare to each other and how hyperparameters influence generalized pooling. We also discuss what these experiments imply for the possibility of learnable generalized pooling.

Five of the pooling methods contain one hyperparameter each. They are $p \in [1, \infty)$ for l_p norm pooling; $\beta \in [0, \infty)$ for soft pooling; $\beta \in [0, \infty)$ for logavgexp pooling; $k \in \{1, 2, ..., N\}$ for k-max pooling, and $\alpha \in [0, 1]$ for mixed pooling. For the pooling benchmark experiment, we select a canonical value for each hyperparameter for simplicity: *i.e.* $p = 2, \beta = 1, \beta = 1, k = 2, \alpha = 0.5$.

The global pooling benchmark results are shown in Table 1. On the left side are results produced by ResNet-50 backbones. The top entry for each dataset is highlighted in bold and the second best is underlined. K-max pooling ($k = 2$) has the best performance across the nine pooling methods we tested, with the highest accuracy across all three datasets, attaining 1.34%, 0.63%, and 0.99% increases over the average pooling baseline for Birds, Cars and Aircraft. These results are actually highly competitive considering the difficulty of fine-grained datasets. If we train k-max pooling ($k = 2$) longer (10^{-3} for 50 epochs, 10^{-4} for 30 epochs and 10^{-5} for 20 epochs), it can reach 87.2% accuracy on the Birds dataset, surpassing all higher-order pooling methods proposed for fine-grained recognition (*e.g.* 86.4% from kernel pooling (Cui et al., 2017), 85.3% from alpha pooling (Simon et al., 2017)). K-max pooling also has very low complexity and memory and computation requirements when compared to the higher-order pooling methods.

The second best method is mixed pooling ($\alpha = 0.5$). It obtained the third best result on the Birds dataset and second best on Cars and Aircraft. It is another highly competitive pooling methods that has been neglected in fine-grained recognition, though it shows larger variance than other algorithms. Max pooling is overall the third best performer, more or less on par with mixed pooling. Nearly all the generalized pooling methods outperform the average pooling baseline. Soft pooling is the worst performer, with several entries below baseline. The overall worst performing method is stochastic pooling, with the lowest accuracy for all three datasets.

On the right side of Table 1 we show results on the Birds dataset using Inception-v3 as the backbone model. The purpose of this experiment is not to optimize for performance, but to compare pooling methods between different backbone models. Max, mixed and k-max pooling still perform strongly in this setting, ranking 2nd, 3rd and 5th respectively. Surprisingly, stochastic pooling attains the best accuracy (83.32%) in this case. This implies that it may not be realistic to find a universal "optimal" pooling scheme across all models and datasets.

These results show directly that average pooling is not the optimal choice for fine-grained recognition. Our careful examination of multiple pooling methods supports our intuition about the impor-

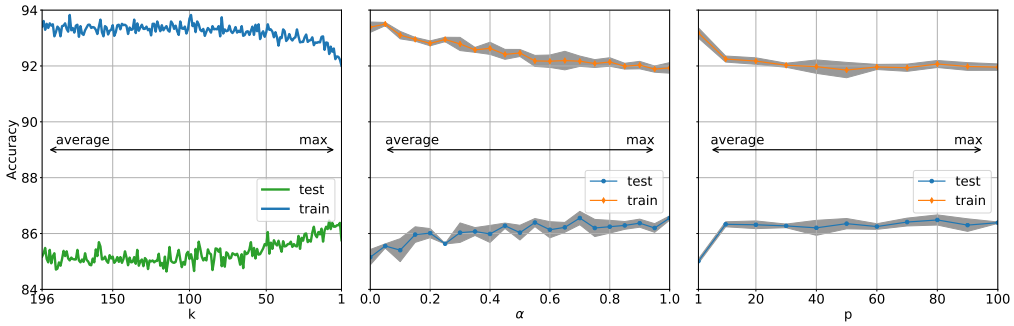

Figure 3: Model accuracy on training and testing set for k-max (left), mixed (middle) and l_p norm (right) pooling with varied k, α and p. Shaded area indicates the standard deviation obtained from multiple runs. Our observation is: max pooling is consistently better than average pooling; model accuracy increases approximately monotonically when generalized pooling changes from average to max pooling; and max pooling generalizes better than average pooling.

Table 1: Mean accuracy and standard deviation are shown for nine pooling schemes on three fine-grained datasets using two backbone models: ResNet-50 and Inception-v3. The best accuracy on each dataset is in bold and the second best is underlined. The best performers are: k-max, mixed, and max pooling. Interestingly, stochastic pooling is the worst performer with ResNet backbone, but ranked the first on Inception-v3.

Pooling Method		**Architecture and Dataset**			
		ResNet			Inception
		Birds	Cars	Airplane	Birds
Average		84.96±0.32	92.29±0.10	88.55±0.22	81.69±0.26
K-max	$k = 2$	**86.30±0.09**	**92.92±0.09**	**89.54±0.18**	82.48±0.15
Mixed	$\alpha = 0.5$	85.97±0.33	<u>92.88±0.16</u>	<u>89.38±0.82</u>	82.88±0.24
Max		<u>86.11±0.26</u>	92.84±0.29	89.14±0.48	<u>82.99±0.19</u>
LogAvgExp	$\beta = 1$	85.85±0.06	92.77±0.11	88.97±0.44	81.87±0.11
L_p	$p = 2$	85.45±0.17	92.55±0.21	88.92±0.18	81.80±0.14
Gated		85.69±0.27	92.35±0.14	88.66±0.26	82.57±0.38
Soft	$\beta = 1$	85.58±0.03	92.20±0.09	87.27±0.17	80.94±0.23
Stochastic		84.74±0.09	91.95±0.28	87.07±0.47	**83.32±0.46**

tance of localized features for fine-grained recognition, and these features aren't captured as well by global average pooling.

Image Resolution Table 1 shows that max pooling outperforms average across datasets and models. These results were obtained using 448×448 input images. To test robustness to image size, we trained 10 models on inputs ranging from 224×224 to 448×448. Results are shown on the right of Figure 2. Max pooling outperforms average pooling by a non-trivial margin for all tested resolutions.

Taken together, the above results validate our first observation:
1. Max pooling performs better than average when fine-tuned from a pretrained model across datasets, models and input resolutions in fine-grained visual categorization.

Hyperparameters Table 1 only includes results for one value of each hyperparameter for the generalized pooling methods. We next evaluate the performance of three generalized pooling with varied parameters. For k-max pooling, we train 196 models, one for each possible value of k. For l_p norm and mixed pooling, we trained 10 and 20 models with linearly spaced parameters. Figure 3 shows the training and testing accuracy. For all three cases, although small perturbations exist, generalized pooling seems to be bounded by the extremes of max and average pooling. As average pooling transforms to max pooling (when k changes from 196 to 1, α from 0 to 1 and p from 1

to 100), the mean accuracy increases nearly monotonically, with few exceptions. If we look at the generalization gap in all three graphs, we can see that max pooling always has lower training accuracy and higher testing accuracy than average pooling.

From these phenomena we make the following observations:
2. Performance increases almost monotonically as average pooling moves to max pooling.
3. Max pooling outperforms average pooling by reducing overfitting.

Learnable Generalized Pooling Our models so far have all involved human-designed pooling, sometimes with a hand-selected hyperparameter. It would be preferred if the pooling function could be *learned*. Generalized pooling provides an intuitive route: a smooth transition between average and max pooling can be achieved if the network learns the optimal hyperparameter. The hope is to learn a "magic" pooling parameter that outperforms both average and max pooling. But learning such a generalized pooling function is complicated by the fact that performance appears to be bounded by max pooling; and more importantly, the learned pooling trained to minimize training loss is likely to to converge to average pooling, which has the lowest training loss and highest training accuracy, leading to overfitting according to Figure 3.

Post-global Batch Normalization Batch normalization (Ioffe & Szegedy, 2015) has been adopted by most popular models to reduce output shift and to help with convergence. It is common practice to add it to the output of convolutional layers; however, we find that adding a batch normalization layer to the output of the global pooling layer is universally beneficial for all models we've tested. It is shown that dependence on "single directions" (single filter outputs) is an indicator of the network's overfitting (Morcos et al., 2018) and we do find in our experiments that the global pooling output actually has a very large variance between filters, indicating a risk of emphasis on single directions. The advantage of post-global batch normalization is two-fold: it helps models with different pooling methods to converge, and it universally improves the performance by $\sim 1\%$. This margin, coupled with Figure 3, indicates that reducing overfitting may be an important topic in fine-grained recognition.

Table 2: Heterogeneous pooling integration results. The offline training method (MLP) generally performs best. Freeze-and-train works best among the end-to-end learnable models.

Dataset	**Pooling Method**						
	Average	Max	MLP	Mixed	Split	Branching	Freeze
Birds	84.96	86.11	**86.66**	85.97	85.49	85.24	<u>86.40</u>
Cars	92.29	92.84	**93.25**	92.88	92.56	91.90	<u>93.08</u>
Aircraft	88.55	89.14	<u>89.62</u>	89.38	88.45	88.72	**89.97**

5.3 Heterogeneous Pooling Integration

In this section, we describe the experiments for heterogeneous pooling integration. We use ResNet-50 as the backbone model. In the offline classifier experiment, we first trained two models with global average and max pooling separately. The two models are then applied to each image as a feature extractor. We repeat this process four times on the training set with varied random seeds as dataset enhancement. A multi-layer perceptron (MLP) is trained using the concatenated features. The MLP we used has 4096 input nodes and 1024 hidden nodes. The number of output nodes equals the number of classes. The learning rate is 10^{-4} for 30 epochs. The offline MLP classifier gets 86.66%, 93.25% and 89.62% respectively on the three fine-grained datasets, generally surpassing all of the single training stage pooling models, the only exception being freeze-and-train on the Aircraft dataset.

For the channel splitting experiment, we use 0.5 as splitting ratio: half of the feature maps are fed into average pooling and the other half into max pooling. The mean accuracies are 85.49%, 92.56% and 88.45%. The branching strategy gets 85.24%, 91.90% and 88.72%. Both strategies are worse than max pooling. Mixed pooling achieves 85.97%, 92.88% and 89.38%, as shown in Table 1. Mixed pooling is better than max pooling on two datasets, and worse than max pooling on one.

For the freeze-and-train method, we modify the branching strategy in the following way: the backbone with global average pooling is trained for 50 epochs and then frozen. The newly added linear layer with global max pooling is then trained for 80 epochs before the whole network is trained for 2 more epochs. This trick guarantees that the final performance won't degrade when the new pooling layer is added. As a result, the final accuracies on the three datasets are 86.40%, 93.08% and 89.97%, surpassing all end-to-end learnable methods and even the offline MLP in the case of the Aircraft dataset.

6 Conclusion and Future Work

In this paper, we focus on the global pooling layer in popular classification models as applied to the task of fine-grained recognition. By visualizing the final conv-layer filters and feature maps, we discover that max pooling produces much sparser feature maps and helps the network learn part-level features. Average pooling, on the other hand, encourages object-level features to be learned. We evaluated nine representative global pooling schemes for fine-grained recognition. K-max ($k = 2$) pooling outperformed all other global pooling schemes and is actually better than all higher-order pooling models. We made several observations from pooling benchmark experiments: (1) max pooling performs better than average pooling across datasets, models, and input resolution; (2) max pooling generalizes better than average pooling; and (3) model performance displays an approximately monotonically increasing characteristic when generalized pooling changes from average to max. Based on these observations, we discussed the potential risk of learning a generalized pooling: namely that minimizing training loss may lead to average pooling and thus be prone to overfitting. We highlight the importance of post-global batch normalization – which is absent from most, if not all, popular state-of-the-art models – in helping to attain faster convergence and in consistently improving model performance. We evaluated several strategies for heterogeneous pooling integration. The freeze-and-train trick performs best among all end-to-end learnable models.

For future work, we suggest consideration of models learned from scratch alongside those fine-tuned from pretrained weights. In addition, experiments should be explored on a broader set of data, not just on fine-grained datasets, in order to affirm whether the findings presented here generalize to more general-purpose datasets such as ImageNet and/or MS-COCO.

References

Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 111–118, 2010.

Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pp. 1–2. Prague, 2004.

Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge Belongie. Kernel Pooling for Convolutional Neural Networks. In *CVPR*, 2017.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. 2005.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Ryan Farrell, Om Oza, Vlad I. Morariu, Ning Zhang, Trevor Darrell, and Larry S. Davis. Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In *ICCV*, 2011. ISBN 978-1-4577-1102-2. doi: 10.1109/ICCV.2011.6126238.

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 4 2007. ISSN 10773142. doi: 10.1016/j.cviu.2005.09.012. URL http://www.sciencedirect.com/science/article/pii/S1077314206001688.

Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact Bilinear Pooling. In *CVPR*, 2016.

Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pp. 2146–2153. IEEE, 2009.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

Jin-Hwa Kim, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Hadamard product for low-rank bilinear pooling. *arXiv preprint arXiv:1610.04325*, 2016.

Jan J Koenderink and Andrea J Van Doorn. The structure of locally orderless images. *International Journal of Computer Vision*, 31(2-3):159–168, 1999.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 2169–2178. IEEE, 2006.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial intelligence and statistics*, pp. 464–472, 2016.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear CNN Models for Fine-Grained Visual Recognition. In *ICCV*, 2015. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.170.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*, 2018.

Naila Murray and Florent Perronnin. Generalized max pooling. In *CVPR*, 2014.

Maria-Elena Nilsback and Andrew Zisserman. Delving deeper into the whorl of flower segmentation. *Image and Vision Computing*, 28(6):1049–1062, 6 2010. ISSN 02628856. doi: 10.1016/j.imavis.2009.10.001.

Eleanor Rosch, Carolyn B Mervis, Wayne D Gray, David M, and Penny Boyes-braem. Basic objects in natural categories. *Cognitive Psychology*, 1976.

Faraz Saeedan, Nicolas Weber, Michael Goesele, and Stefan Roth. Detail-preserving pooling in deep networks. In *CVPR*, 2018.

Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pp. 92–101. Springer, 2010.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshops*, 2014.

Marcel Simon, Yang Gao, Trevor Darrell, Joachim Denzler, and Erik Rodner. Generalized orderless pooling performs implicit salient matching. In *ICCV*, 2017.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015.

C Szegedy, Wei Liu, Yangqing Jia, P Sermanet, S Reed, D Anguelov, D Erhan, V Vanhoucke, and A Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. doi: 10.1109/CVPR.2015.7298594.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, 2016.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.

Yaming Wang, Vlad I Morariu, and Larry S Davis. Learning a discriminative filter bank within a cnn for fine-grained recognition. In *CVPR*, 2018.

Xing Wei, Yue Zhang, Yihong Gong, Jiawei Zhang, and Nanning Zheng. Grassmann pooling as compact homogeneous bilinear pooling for fine-grained visual classification. In *ECCV*, 2018.

M. D Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. *ECCV*, 2013.

Matthew D Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013.

Shuangfei Zhai, Hui Wu, Abhishek Kumar, Yu Cheng, Yongxi Lu, Zhongfei Zhang, and Rogerio Feris. S3pool: Pooling with stochastic spatial sampling. In *CVPR*, 2017.

Richard Zhang. Making convolutional networks shift-invariant again. *arXiv preprint arXiv:1904.11486*, 2019.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.

Feng Zhou and Yuanqing Lin. Fine-Grained Image Classification by Exploring Bipartite-Graph Labels. In *CVPR*, 2016.